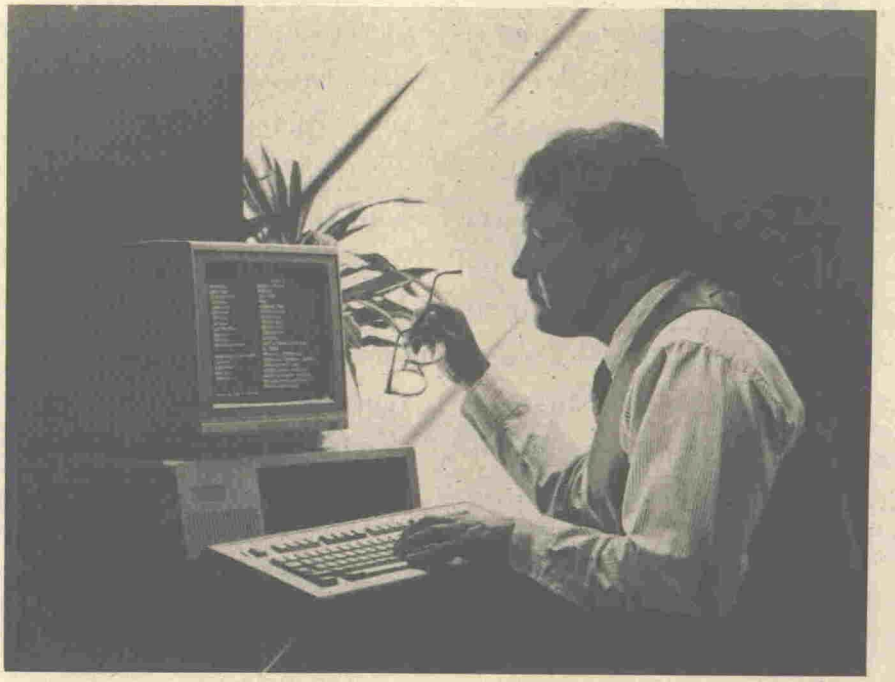
วีรวัจน์ บัวทอง

SOFT MAIL

เรียกใช้งานโปรแกรมสะดวกขึ้น ด้วย UI

ยูทิลิตี้ช่วยเรียกใช้งานโปรแกรมที่เขียนขึ้นด้วยภาษาโปรล็อก



มีโปรแกรมอยู่หลายโปรแกรมที่มีความสามารถสูง แต่ไม่ค่อยมีคนใช้แพร่หลาย เพราะใช้งานลำบาก เวลาใช้ต้องจำฟอร์แมตการเรียกใช้ให้ได้ ทั้งยังมี option switch (คือ /w /t เป็นต้น) อยู่หลายอย่าง ผู้เขียนจึงเกิดความคิดที่จะเขียนโปรแกรมช่วยให้การเรียกใช้โปรแกรมเหล่านั้นมาใช้งานได้ง่ายขึ้น โดยนำตัวอย่างมาจากโปรแกรม NI-Norton Integrater ที่อยู่ในชุดของ Norton Utilities นั่นเอง

ลักษณะของโปรแกรม

โปรแกรม NI หรือโปรแกรม UI-Utility Integrater ที่ผู้เขียนนำมาเขียนใหม่ (ที่จริงคือลอกแบบมา) นี้ ประกอบด้วยวินโดว์ด้านซ้ายด้านขวาและด้านล่าง วินโดว์ด้านซ้ายคือ menu window ซึ่งจะแสดงชื่อของโปรแกรมภายนอกต่าง ๆ ที่เราสามารถเรียกใช้งานได้ส่วนวินโดว์ด้านขวา คือ hint window ที่จะแสดงฟอร์แมตการเรียกใช้งานพร้อมด้วยความหมายของ option switch ต่าง ๆ ส่วนวินโดว์ด้านล่างคือ command window เป็นส่วนที่เราจะ key parameters ต่าง ๆ เข้าไป (ดูรูปที่ 1)

โปรแกรม UI นี้ ประกอบด้วยไฟล์ทั้งหมด 6 ไฟล์ดังนี้

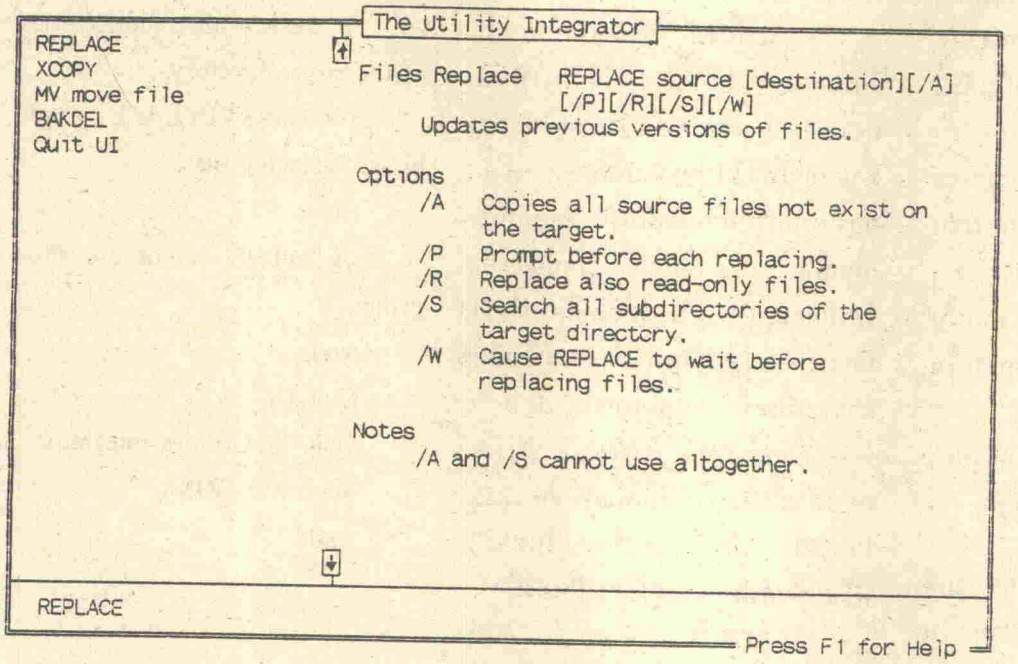
1. UI.PRO คือโปรแกรมหลัก
2. UIINIT.PRO คือโปรแกรมที่เป็น predicate ที่สำคัญต่าง ๆ
3. LONGMEN1.PRO คือโปรแกรมในส่วนของ predicate longmenu
4. UISCREEN คือส่วนที่เป็น background ของจอ ขณะที่รันโปรแกรมนี้
5. HELP.UI คือส่วนที่เป็น help file
6. HINT.UI คือส่วนที่เก็บคำแนะนำในการใช้โปรแกรมต่าง ๆ

สามไฟล์แรกเมื่อถูกคอมพายล์จะกลายเป็นไฟล์ UI.EXE ส่วน 3 ไฟล์หลังจะถูกเรียกใช้โดยไฟล์ UI.EXE ดังนั้นไฟล์ทั้งหมดเหล่านี้จะต้องอยู่ในไดเรคทอรีเดียวกัน

ส่วนประกอบของโปรแกรมที่สำคัญ

1. ส่วนที่ Initial โปรแกรม ทำหน้าที่ detect ชนิดของจอภาพ และสร้างวินโดว์ทั้งหลาย

2. ส่วนที่สร้างเมนู คือส่วนที่แสดงรายชื่อของโปรแกรมภายนอกต่าง ๆ ไว้ในลักษณะเมนูในจอภาพด้านซ้ายมือ ประกอบด้วยพรีดิเคตหลักชื่อ longmenu พรีดิเคตนี้ผู้เขียนได้นำพรีดิเคต



รูปที่ 1 โฉมหน้าของโปรแกรม UI

ที่อยู่ใน Toolbox ของบริษัท Borland International มาดัดแปลง โดยการเพิ่ม predicate write__command__hint ซึ่งจะ write ชื่อโปรแกรมภายนอกไว้ใน command window และแสดงคำแนะนำการใช้โปรแกรมนั้น ๆ ไว้ใน hint window ด้านขวามือ พร้อมทั้งดัดแปลงใน predicate longmenu2 ให้สามารถใส่ command line parameter ต่าง ๆ เข้าไปได้ด้วย

3. ส่วนที่สร้าง Popup window ทำหน้าที่แสดงวินโดว์แบบ explode เมื่อมีการกดคีย์ <F1> (help) เพื่อช่วยให้โปรแกรมนี้ดูน่าสนใจขึ้น ประกอบด้วยพรีดิเคตหลักชื่อ popup__window ทำหน้าที่สร้างวินโดว์ขึ้นทีละอันซ้อน ๆ กันจนได้ขนาดตามต้องการ โดยที่ระยะเวลาระหว่างการ popup ของวินโดว์นั้น กำหนดได้โดยใช้พรีดิเคต delay

4. ส่วนที่สร้าง Hint message คือส่วนที่อ่านคำแนะนำการใช้โปรแกรมภายนอกต่าง ๆ จาก ไฟล์ "HINT.UI" มาเก็บไว้ในดาต้าเบสชื่อ UI__hint โดยมีพรีดิเคต hint ทำหน้าที่เรียกข้อมูลในดาต้าเบสนี้ไปแสดงใน hint window

5. ส่วนที่รวบรวมรายชื่อของโปรแกรมภายนอก เพื่อใช้สร้างเป็น choice ของ command menu มีพรีดิเคต get__choice เป็นพรีดิเคตหลัก ทำหน้าที่รวบรวมรายชื่อโปรแกรมภายนอกต่าง ๆ ที่อยู่ในดาต้าเบส UI__hint

6. ส่วนที่สร้าง Help window ประกอบด้วย 2 ส่วน ส่วนแรกคือพรีดิเคต init__help จะอ่านไฟล์ "HELP.UI" เข้ามาเก็บในดาต้าเบสชื่อ UI__help ส่วนที่ 2 จะเอาข้อมูลจาก UI__help ไปแสดงผล เมื่อมีการกดคีย์ <F1>

7. ส่วนที่สร้าง Scroll bar เพื่อใช้แสดงให้รู้ว่า ยังมีรายชื่อโปรแกรมภายนอกอยู่อีกหรือไม่ (ส่วนนี้เป็นส่วนที่ผู้เขียนดัดแปลงให้แตกต่างไปจากโปรแกรม NI เพราะชอบวิธีนี้มากกว่า)

8. ส่วนที่ execute โปรแกรมภายนอก คือส่วนที่จะรันโปรแกรมที่เลือก จากนั้นจะรอให้กดคีย์ใด ๆ เพื่อกลับเข้าสู่โปรแกรม UI ประกอบด้วยพรีดิเคตหลัก คือ พรีดิเคต do__command และพรีดิเคต return

ไฟล์ HINT.UI

คือไฟล์ที่เก็บคำแนะนำการใช้งานของแต่ละโปรแกรมภายนอกที่เราต้องการ ดังนั้นผู้ใช้แต่ละรายจึงต้องสร้างไฟล์นี้ขึ้นมาเองโดยใช้เวิร์ดโปรเซสเซอร์ที่เป็น ASCII text เท่านั้น ซึ่งถ้าจะให้ดี ควรใช้ editor ของ prolog เอง เพื่อหลีกเลี่ยงปัญหาไม่คาดคิดที่อาจเกิดขึ้นได้ ในขั้นแรกที่จะสร้างไฟล์นี้นั้น

จะต้องเลือกก่อนว่า มีโปรแกรมอะไรบ้าง ที่เวลาจะใช้งาน เราจะต้องไปเปิดดูคู่มือเป็นประจำ (เช่นโปรแกรม XCOPY หรือ REPLACE ของดอสเป็นต้น) หรือไม่ก็ต้องลองรันโปรแกรมนั้นดู ถ้าเรียกใช้ไม่ถูกต้อง บางโปรแกรมก็จะบอกฟอร์แมตที่ถูกต้องมาให้ทราบ (เช่นโปรแกรม Touch หรือโปรแกรมต่าง ๆ ใน Norton Utilities เป็นต้น) จากนั้นก็พิมพ์วิธีการใช้งานของโปรแกรมนั้นเข้าไป

สำหรับข้อกำหนดอื่นที่สำคัญของการพิมพ์ข้อมูลเข้าไปในไฟล์นี้ก็คือ

1. คำแรกในบรรทัดแรกของแต่ละโปรแกรมจะต้องเป็นชื่อโปรแกรมเท่านั้น จากนั้นเราอาจจะพิมพ์ข้อความเข้าไปข้างหลังเพื่อขยายความก็ได้ แต่ต้องมีช่องว่างระหว่างชื่อโปรแกรม และคำขยายความนั้นและความยาวของบรรทัดแรกนี้รวมทั้งหมดจะต้องไม่เกิน 25 ตัวอักษร เพราะข้อความทั้งหมดในบรรทัดนี้จะถูก read ไปเป็น choice ของ command menu ตัวอย่างเช่น ถ้าโปรแกรมชื่อ NCD บรรทัดแรกจะพิมพ์ว่า NCD หรือ NCD change directory เพื่อช่วยขยายความว่า โปรแกรมนี้ใช้สำหรับเปลี่ยนไดเรคทอรีก็ได้

2. ตั้งแต่บรรทัดที่ 2 จนถึงบรรทัดที่ 20 คือคำแนะนำที่เราต้องการให้ปรากฏใน hint window โดยบรรทัดต้น ๆ ควรเป็นฟอร์แมตการใช้งาน จากนั้นจึงเป็น option switch ต่าง ๆ และหมายเหตุ (ถ้ามี) ความยาวของแต่ละจะต้องไม่เกิน 52 ตัวอักษร ข้อความที่เกินกว่านั้นจะไม่สามารถปรากฏในวินโดว์ได้

3. บรรทัดที่ 21 จะต้องมีคำว่า :end: ในคอลัมน์ที่ 1 เพื่อเป็น keyword ที่ให้โปรแกรม UI รู้ว่า จบคำแนะนำสำหรับโปรแกรมนั้น ๆ แล้ว

4. จะต้องมี คำสั่งให้ออกจากโปรแกรม UI นี้ด้วย โดยการพิมพ์คำว่า Quit หรือ Quit UI อยู่ในตอนท้ายของไฟล์ในบรรทัดต่อจาก :end: ของโปรแกรมภายนอกโปรแกรมสุดท้าย เพื่อเป็น choice ให้เราสามารถออกจากโปรแกรม UI นี้ได้ เมื่อใดก็ตามที่เรามีการเพิ่มโปรแกรมภายนอกเข้าไปอีก ควรจะเพิ่มเข้าไปก่อนคำสั่ง Quit นี้

ตัวอย่างไฟล์ HINT.UI ที่ประกอบด้วยโปรแกรมภายนอก ชื่อ REPLACE (เป็นโปรแกรมที่อยู่ในดอส) พร้อมทั้งคำสั่ง QUIT จะเป็นดังนี้

REPLACE

Files Replace REPLACE source [destination][/A][/P][/R][/S][/W]

Updates previous versions of files.

Options

/A Copies all source files not exist on the target.

/P Prompt before each replacing.

/R Replace also read-only files.

/S Search all subdirectories of the target directory.

/W Cause REPLACE to wait before replacing files.

Notes

/A and /S cannot use altogether.

:end:

Quit UI

Quit the Utilities Integrator

Return to DOS

:end:

ไฟล์ HELP.UI

สำหรับผู้อ่านที่ชอบการพิมพ์โปรแกรมด้วยตนเอง จะต้องสร้างไฟล์ HELP.UI ขึ้นมาด้วย ไม่เช่นนั้นจะเกิด runtime error ขึ้น help file ที่ผู้เขียนสร้างขึ้นนี้ มีขนาดกว้าง 68 ตัวอักษร ยาว 20 บรรทัด ไฟล์นี้จะแสดงให้ผู้ใช้ทราบถึงการใช้คีย์ต่าง ๆ (ดูรูปที่ 2)

```
                The Utility Integrator
                Standard Edition 1.0
            Copyright (C) January 1991
            Written by Werawat Buatong

You can execute many of utilities from within UI.  Move the
highlight to the command, type any arguments, and press Enter.


Enter          Run the program on the command line
Home           Move to first command
End            Move to last command
↑ or ↓         Move to previous or next command
Ins            Switch on/off insert mode
Letters        Type arguments on the command line
Esc, F10       Quit UI
```

รูปที่ 2 ไฟล์ HELP.UI

พรีดิเคตที่สำคัญ

เนื่องจากโปรแกรมนี้ค่อนข้างยาว และประกอบด้วยพรีดิเคตจำนวนมาก ผู้เขียนจึงไม่อาจจะอธิบายรายละเอียดของพรีดิเคตต่าง ๆ ได้หมด เพราะจะเป็นการสิ้นเปลืองเนื้อที่จนเกินไป จึงขออธิบายเฉพาะพรีดิเคตที่เห็นว่าเป็นประโยชน์เท่านั้น ซึ่งได้แก่พรีดิเคต longmenu พรีดิเคต scroll_bar และพรีดิเคต return

พรีดิเคต longmenu

พรีดิเคตนี้เป็นหัวใจสำคัญของโปรแกรมนี้ ความหมายของคำว่า longmenu หมายถึงเมนูที่มี choice อยู่มากกว่าความยาวของวินโดว์ เช่น มีอยู่ 8 choice ในขณะที่วินโดว์ยาวเพียง 5 บรรทัด ดังนั้นปกติแล้ว choice ที่ 6 ถึง 8 จะมองไม่เห็น ต่อเมื่อเราเลื่อนเคอร์เซอร์ลงไป choice เหล่านั้นจึง scroll ขึ้นมา (ในขณะที่ choice ต้น ๆ ก็จะหายไป เพราะความยาวของวินโดว์มีจำกัดนั้นเอง)

สำหรับในโปรแกรมนี้ มีสิ่งที่จะต้องไปจาก longmenu ที่มีมากับ Toolbox ตามที่ได้กล่าวไว้ตอนต้นแล้ว สำหรับ listing ของพรีดิเคตนี้ที่ได้ดัดแปลงแล้วจะอยู่ในไฟล์ LONGMEN1.PRO

พรีดิเคต Scroll_bar

เป็นพรีดิเคตที่ผู้เขียนเห็นว่าสามารถนำไปใช้ในโปรแกรมอื่นได้มาก พรีดิเคตนี้ใช้สำหรับสร้าง scroll bar ในแนวตั้ง เพื่อเป็นสัญลักษณ์ที่ใช้แสดงให้ทราบว่า ขณะนี้เราอยู่ที่ส่วนใดของเมนูโดยมี box เล็ก ๆ แสดงให้ทราบถึงตำแหน่ง เช่น ถ้าเราอยู่ที่ choice แรก box นี้ก็จะอยู่ส่วนบนสุดของ scroll bar เป็นต้น (scroll bar นี้จะหาดูได้จากโปรแกรมของบริษัทไมโครซอฟต์ทั้งหลาย)

สำหรับฟอร์แมตการเรียกใช้พรีดิเคตจะเป็นดังนี้

scroll_bar(Row,Column,Height)
/* (i,i,i) */

โดยที่ Row และ Column คือตำแหน่งในจอภาพ และ Height คือความยาวของ scroll bar ซึ่งจะต้อง bond ทั้ง 3 ค่า เมื่อเราต้องการให้แสดง box ที่ตำแหน่งใด ก็จะต้องเรียกไปยังพรีดิเคต scr_box(add) พร้อมทั้งคำนวณหาค่า N แล้วเรียกไปยังพรีดิเคต scroll_box(N) และเมื่อใดที่ต้องการเลื่อน box นี้ อย่าลืมว่าเราจะต้องลบ box เก่าออกก่อนเสมอ โดยการเรียกไปยังพรีดิเคต scr_box(remove)

พรีดิเคต return

ก่อนหน้าที่จะเขียนโปรแกรมนี้ขึ้นมา ผู้เขียนเคยพบปัญหาเกี่ยวกับตำแหน่งของเคอร์เซอร์หลังจากที่มีการเรียกใช้พรีดิเคต system เพราะว่าหลังจากเรียกแล้ว ตำแหน่งของเคอร์เซอร์แทนที่จะไปอยู่บรรทัดว่าง ๆ ตอนท้ายปรากฏว่ามักจะไปทับอยู่บนบรรทัดที่มีตัวอักษรเสมอ ๆ ดังนั้นเมื่อมีการ write ข้อความใด ๆ (ซึ่งในกรณีนี้คือ "press any key to continue...") ข้อความนั้นจะไปทับกับข้อความเดิมบนจอภาพ ทำให้อ่านไม่รู้เรื่อง ผู้เขียนพยายามจะหาวิธีแก้แต่ก็ไม่สำเร็จ เพิ่งมาคิดวิธีการหลีกเลี่ยง (ไม่ใช่แก้ไข) ได้ตอนที่ทำโปรแกรมนี้ วิธีการก็ไม่มีอะไรมากเพียงแต่ตรวจสอบตั้งแต่บรรทัดที่ 24 ว่า มีข้อความอะไรแสดงอยู่หรือไม่ ถ้าไม่มีก็เลื่อนไปตรวจสอบบรรทัดก่อนหน้า จนกว่าจะพบบรรทัดที่ว่าง เมื่อพบแล้วจึงย้ายเคอร์เซอร์ไปไว้ที่ตำแหน่งนั้น

เขียนเป็นภาษา prolog ได้ดังนี้

```
.....
field_str(N,0,80,String),
% String=ข้อความในคอลัมน์ 0-80 บรรทัด N
not(fronttoken(String,_,_)),
% ถ้า String=""ให้ตรวจสอบบรรทัดถัดขึ้นไป
```

listing ทั้งหมดของพรีดิเคตจะอยู่ในไฟล์ UI.PRO

การนำโปรแกรมไปใช้งาน

หลังจากที่คีย์โปรแกรมเสร็จสิ้นแล้วและคอมพายล์แล้วจะได้เป็นไฟล์ชื่อ UI.EXE ให้ทดลองรันดู ถ้ารันแล้วเกิดปัญหาขึ้นอาจจะเป็นเพราะว่าโปรแกรมหาไฟล์ที่ 4-6 ที่กล่าวไว้ในตอนต้นไม่พบ ให้ก๊อบปี้มาไว้ที่ไดเรคทอรีเดียวกันกับไฟล์ UI.EXE ถ้ารันแล้วยังมี error อีกอาจเป็นเพราะสร้างไฟล์ HINT.UI ไม่ถูกต้อง เป็นต้น

เมื่อใดก็ตามที่เราต้องการเพิ่มรายชื่อของโปรแกรมภายนอกเข้าไป ก็สามารถทำได้โดยการเพิ่มชื่อและคำแนะนำการใช้โปรแกรมนั้น ๆ เข้าไปในไฟล์ HINT.UI ได้โดยตรงตามรูปแบบที่ได้กล่าวไว้แล้วและเพราะเนื่องจากไฟล์ HINT.UI เป็นข้อมูลภายนอกที่จะถูกโปรแกรมหลักอ่านเข้าไป ไม่ได้มีส่วนเกี่ยวข้องในการ flow ของโปรแกรมเลย เราจึงไม่ต้องคอมพายล์โปรแกรมใหม่ใด ๆ ทั้งสิ้น

เพื่อให้ใช้งานได้สะดวก ควรจะสร้าง path สำหรับ โปรแกรมนี้ และโปรแกรมภายนอกทั้งหมดด้วย เพราะเมื่อไรก็ตามที่เราจำฟอร์แมตการเรียกโปรแกรมภายนอกได้แล้ว ก็ไม่มีความจำเป็นที่จะต้องเรียกโปรแกรมนี้แต่อย่างใด (ในกรณีนี้ เราจะต้องระบุ path ของไฟล์ UISCREEN HELP.UI และ HINT.UI ไว้ในโปรแกรมทุกแห่งที่มีการอ่านไฟล์เหล่านี้ด้วย)

ผู้เขียนหวังเป็นอย่างยิ่งว่า โปรแกรมนี้จะช่วยให้ท่านสามารถใช้โปรแกรมจำพวกดอสได้อย่างสะดวกรวดเร็วยิ่งขึ้น โดยไม่ต้องไปเสียเวลาเปิดหนังสือคู่มืออีกต่อไป

หมายเหตุ โปรแกรม UI.PRO, UIINIT.PRO, และ LONGMEN1.PRO แสดงดังรูปที่ 3, 4 และ 5 ตามลำดับ

```
include "uiinit.pro"
include "longmen1.pro"

predicates
   menu_cancel(integer)
   do_command(string)
   return(integer)

goal
   retractall(_),
   detect_crt,assert(d_col(0,maxcol)),
   assert(insmode),
   make_background,
   init_help,
   init_hint,read_hint,
   init_main_screen,
   init_command_window,
   scroll_bar(1,26,21),
   get_choice(L,[]),
   repeat,
   shiftwindow(init_window),
   shiftwindow(hint_window),
   longmenu(1,1,21,scr_color,0,L,"",1,Choice),
   menu_cancel(Choice),
   shiftwindow(command_window),
   window_str(String),
   do_command(String).

clauses
   do_command(String):-
      concat(String1,"\n",String),!,
      do_command(String1).
   do_command(String):-
      fronttoken(String,"Quit",_),!,
      menu_cancel(0).
   do_command(String):-
      shiftwindow(back_window),clearwindow,
      system(String,0,_),
      return(24).

   return(N):-
      N>1,field_str(N,0,80,String),
      not(fronttoken(String,_,_)),!,
      N1=N-1,return(N1).
   return(N):-
      N<23,!,N1=N+2,
      cursor(N1,0),write("press any key to continue..."),
      readkey(_),fail.
   return(_):-
      cursor(24,0),write("\npress any key to continue..."),
      readkey(_),fail.

   menu_cancel(0):-!,
      shiftwindow(back_window),clearwindow,
      write("UI-The Utility Integrater, Standard Edition 1.0,"),
      write(" (C) Copyright January 1991\n"),exit.
   menu_cancel(_).
```

รูปที่ 3 โปรแกรม UI.PRO

รูปที่ 4 โปรแกรม UIINIT.PRO

```
domains
   row, col, attr   = integer
   stringlist = string*
   integerlist = integer*
   key    = cr; esc; break; tab; btab; del; bdel; ctrlbdel; ins;
            end ; home ; fkey(integer) ; up ; down ; left ; right ;
            ctrlleft; ctrlright; ctrlend; ctrlhome; pgup; pgdn;
            ctrlpgup; ctrlpgdn; char(char) ; otherspec

   file=hint_file

database
   determ crt(symbol)
   determ step_height_width(integer,integer)
   determ ui_help(string)
   determ d_col(col,col)
   determ insmode
   determ scroll_b(integer)
   determ total_command(integer)
   determ current_display(integer)
   hint_text(symbol)
   ui_hint(symbol,symbol)

constants
   back_window    = 20
   init_window    = 21
   command_window = 22
   hint_window    = 23
   menu_window    = 24
   help_window    = 25
   scbar_window   = 26

   hilite_attr    = 30
   scr_color      = 31
   inv_attr       = 48
   scr_rev        = 56
   maxcol         = 74
   arrow_color    = 113
   help_color     = 121

/*************************************************************************/
/*                    General predicates                                 */
/*************************************************************************/

predicates
   listlen(stringlist,row)              /* The length of a list */
   member_number(integer,stringlist,string)
   min(row,row,row)
   nondeterm repeat

clauses
   listlen([],0).
   listlen([_|T],N):-
      listlen(T,X),
   N=X+1.

member_number(No,[H|_],Name):-
   No=1,!,Name=H.
member_number(No,[_|T],Name):-
   No1=No-1,member_number(No1,T,Name).

min(X,Y,X):-X<=Y,!.
min(_,X,X).

repeat.
repeat:-repeat.
```

```
/*************************************************************/
/*                    Detect CRT Type                        */
/*************************************************************/

predicates
   detect_crt
   crt_type(integer,symbol)
   set_attr(symbol,symbol,integer,integer)

clauses
   detect_crt:-
      retractall(crt(_)),
      detectgraph(A,_),
      crt_type(A,Crt),
      assert(crt(Crt)).

   crt_type(3,"ega"):-!.
   crt_type(7,"hercmono"):-!.
   crt_type(9,"vga").

   set_attr(hercmono,reverse,_,112):-!.
   set_attr(_,reverse,scr_color,scr_rev):-!.
   set_attr(_,_,Color,Color).

/*************************************************************/
/*                  Screen initialization                    */
/*************************************************************/

predicates
   make_background
   init_command_window
   init_main_screen

clauses
   make_background:-
      makewindow(back_window,31,0,"",0,0,25,80).

   init_command_window:-
      crt(Crt),!,set_attr(Crt,reverse,scr_color,Inv_attr),
      field_attr(0,28,24,Inv_attr),
      makewindow(command_window,Inv_attr,0,"",23,3,1,maxcol).
   init_main_screen:-
      makewindow(init_window,31,0,"",0,0,25,80),
      file_str("uiscreen",Screen),
      window_str(Screen).

/*************************************************************/
/*                         Readkey                           */
/*          Returns a symbolic key from the key domain       */
/*************************************************************/

predicates
   readkey(key)
   readkey1(key,char,integer)
   readkey2(key,integer)

clauses
   readkey(KEY):-readchar(T),char_int(T,VAL),readkey1(KEY,T,VAL).

   readkey1(KEY,_,0):-!,readchar(T),char_int(T,VAL),readkey2(KEY,VAL).
   readkey1(cr,_,13):-!.
   readkey1(esc,_,27):-!.
   readkey1(break,_,3):-!.
   readkey1(tab,_,9):-!.
   readkey1(bdel,_,8):-!.
   readkey1(ctrlbdel,_,127):-!.
   readkey1(char(T),T,_) .
```

```
  readkey2(btab,15):-!.
  readkey2(del,83):-!.
  readkey2(ins,82):-!.
  readkey2(up,72):-!.
  readkey2(down,80):-!.
  readkey2(left,75):-!.
  readkey2(right,77):-!.
  readkey2(pgup,73):-!.
  readkey2(pgdn,81):-!.
  readkey2(end,79):-!.
  readkey2(home,71):-!.
  readkey2(ctrlleft,115):-!.
  readkey2(ctrlright,116):-!.
  readkey2(ctrlend,117):-!.
  readkey2(ctrlpgdn,118):-!.
  readkey2(ctrlhome,119):-!.
  readkey2(ctrlpgup,132):-!.
  readkey2(fkey(N),VAL):- VAL>58, VAL<70, N=VAL-58, !.
  readkey2(fkey(N),VAL):- VAL>=84, VAL<104, N=11+VAL-84, !.
  readkey2(otherspec,_).

/*************************************************************/
/*                  Make exploded window                     */
/*************************************************************/

predicates
  delay(integer)
  popup_window(integer,integer,integer,string,integer,integer,integer,integer)
  popup_window1(integer,integer,integer,string,integer,integer,integer,integer)

clauses
  delay(0):-!.
  delay(N):-N>0,!,N1=N-1,delay(N1).

  popup_window(TotalWin,WinNo,Color,WinName,Row,Col,Height,Width):-
    Step_H=Height/TotalWin,Step_W=Width/TotalWin,
    assert(step_height_width(Step_H,Step_W)),
    R=Row+TotalWin*Step_H/2,C=Col+TotalWin*Step_W/2,
    H=Height-TotalWin*Step_H+1,W=Width-TotalWin*Step_W+1,
    popup_window1(TotalWin,WinNo,Color,WinName,R,C,H,W).

  popup_window1(0,WinNo,Color,Name,_,_,_,_):-!,
    delay(1000),
    makewindow(WinNo,Color,0,Name,2,4,20,71).
  popup_window1(TotalWin,WinNo,Color,Name,Row,Col,Height,Width):-
    TotalWin>0,
    step_height_width(H,W),!,
    delay(1000),
    makewindow(WinNo,Color,0,"",Row,Col,Height,Width),
    TotalWin1=TotalWin-1,WinNo1=WinNo+1,Row1=Row-H/2,Col1=Col-W/2,
    Height1=Height+H,Width1=Width+W,
    popup_window1(TotalWin1,WinNo1,Color,Name,Row1,Col1,Height1,Width1).

/*************************************************************/
/*              Split Front Line Of String                   */
/*************************************************************/

predicates
  frontln(string,string,string,string)
clauses
  frontln(String,Front,Rest_string,Init):-
    frontchar(String,Frontchar,Rest),
    Frontchar<>'\n',!,
    str_char(Frontstr,Frontchar),
    concat(Init,Frontstr,New_init),
    Frontln(Rest,Front,Rest_string,New_init).
  frontln(String,Front,String,Front).
```

```
/*************************************************************/
/*                    Make Hint Message                      */
/*************************************************************/

predicates
    init_hint
    read_hint
    check_end(string)
    collect_text(string,string)
    hint(symbol)
    nondeterm repeat_file(file)
clauses
    init_hint:-
        makewindow(hint_window,scr_color,0,"",1,27,21,52),
        openread(hint_file,"HINT.UI"),readdevice(hint_file).

    read_hint:-
        repeat_file(hint_file),readln(Line),check_end(Line),
        concat(Line,"\n",Nline),assert(hint_text(Nline)),fail.
    read_hint:-readdevice(keyboard).

    check_end(Text):-Text<>":end:",!.
    check_end(_):-collect_text("",_).

    collect_text(Init,Text):-
        hint_text(Text0),retract(hint_text(Text0)),!,
        concat(Init,Text0,Text1),collect_text(Text1,Text).
    collect_text(Text,_):-
        frontln(Text,Front,Rest,""),assert(ui_hint(Front,Rest)),fail.

    hint(Command):-
        shiftwindow(Current),
        shiftwindow(hint_window),clearwindow,
        ui_hint(Command,Hint_text),!,window_str(Hint_text),
        shiftwindow(Current).

    repeat_file(_).
    repeat_file(File):-not(eof(File)),repeat_file(File).

/*************************************************************/
/*                 Get Pulldown Menu Choice                  */
/*************************************************************/

predicates
    member(string,stringlist)
    append(stringlist,stringlist,stringlist)
    get_choice(stringlist,stringlist)

clauses
    get_choice(L,Init_list):-
        ui_hint(Choice,_),not(member(Choice,Init_list)),!,
        append(Init_list,[Choice],New_list),
        get_choice(L,New_list).
    get_choice(List,List).

    member(X,[X|_]):-!.
    member(X,[_|Y]):-member(X,Y).

    append([],List,List).
    append([X|L1],List2,[X|L3]):-append(L1,List2,L3).

/*************************************************************/
/*                     Make Help Window                      */
/*************************************************************/
```

```
predicates
   init_help
   help

clauses
   init_help:-
      file_str("HELP.UI",Help_text),
      assert(ui_help(Help_text)).

   help:-crt(Crt),set_attr(Crt,reverse,help_color,Inv_attr),
      popup_window(10,90,Inv_attr,"",2,2,19,68),
      ui_help(Help_text),!,
      window_str(Help_text),readkey(_),
      removewindow,removewindow,
      removewindow,removewindow,removewindow,
      removewindow,removewindow,removewindow,
      removewindow,removewindow,removewindow.

/*************************************************************/
/*                    Make Scroll Bar                        */
/*************************************************************/

predicates
   scroll_bar(integer,integer,integer)
   scroll_box(integer)
   scr_box(symbol)
   replicate(char,integer)

clauses
   scroll_bar(Row,Column,Height):-
      crt(Crt),set_attr(Crt,reverse,arrow_color,Rev),
      shiftwindow(Old),
      makewindow(scbar_window,scr_color,0,"",Row,Column,Height,1),
      H1=Height-1,
      replicate('\176',H1),
      scr_char(0,0,'\24'),scr_attr(0,0,Rev),
      scr_char(H1,0,'\25'),scr_attr(H1,0,Rev),
      shiftwindow(Old).

   scr_box(add):-
      current_display(1),
      assert(scroll_b(1)),
      scroll_box(1),!.
   scr_box(add):-
      current_display(No),total_command(T),
      N=18*(No-1) div (T-1)+1,assert(scroll_b(N)),
      scroll_box(N),!.
   scr_box(remove):-
      shiftwindow(Old),
      shiftwindow(scbar_window),
      scroll_b(N),retract(scroll_b(N)),
      scr_char(N,0,'\176'),
      shiftwindow(Old),!.
   scroll_box(Line):-
      shiftwindow(Old),
      shiftwindow(scbar_window),
      scr_char(Line,0,'\178'),
      shiftwindow(Old).

   replicate(Char,Count):-
      Count>0,!,scr_char(Count,0,Char),
      C1=Count-1,replicate(Char,C1).
   replicate(_,_).
```

รูปที่ 5 โปรแกรม LONGMEN1.PRO

```
predicates
  longmenu(row,col,row,attr,attr,stringlist,string,integer,integer)
  longmenuinit(row,col,row,attr,attr,stringlist,string,row,col,row)
  longmenu1(symbol,row,col,row,col,attr,stringlist,row,row,row,row,row,row)
  longmenu2(symbol,row,row,row,row,row,row,key)
  longmenu3(row,row,row,row,row,row)
  wr_part_if_changed(row,row,row,col,stringlist)
  write_part_list(row,row,row,row,col,stringlist)
  write_command_hint(integer,stringlist)
  changemode

clauses
  longmenu(ROW,COL,MAXH,WATTR,FATTR,STRINGLIST,HEADER,STCHOICE,CHOICE) :-
        longmenuinit(ROW,COL,MAXH,WATTR,FATTR,STRINGLIST,HEADER,HEIGHT,LEN,NOOFROW),
        STOFFSET=STCHOICE-1,
        longmenu3(NOOFROW,HEIGHT,0,STOFFSET,BASE,OFFSET),
        No=BASE+OFFSET+1,write_command_hint(No,STRINGLIST),
        longmenu1(cont,ROW,COL,HEIGHT,LEN,WATTR,STRINGLIST,NOOFROW,-1,BASE,OFFSET,BASE1,OF
        CHOICE = BASE1 + OFFS1+1,
        removewindow.

  longmenuinit(ROW,COL,MAXH,WATTR,FATTR,STRINGLIST,HEADER,HEIGHT,NOOFCOL,NOOFROW):-
        NOOFCOL=25,
        listlen(STRINGLIST,N), N > 0 , NOOFROW=N,
        assert(total_command(N)),
        min(NOOFROW,MAXH,HEIGHT),
        makewindow(menu_window,WATTR,FATTR,HEADER,ROW,COL,HEIGHT,NOOFCOL).

  write_command_hint(No,STRINGLIST):-
        not(current_display(No)),
        retractall(current_display(_)),
        assert(current_display(No)),
        member_number(No,STRINGLIST,Name),
        fronttoken(Name,Front,_),str_len(Front,Len),
        d_col(Min,Max),retract(d_col(Min,Max)),assert(d_col(Len,Max)),
        hint(Name),!,
        shiftwindow(command_window),clearwindow,
        concat(Front," ",F),
        write(F),shiftwindow(menu_window).
  write_command_hint(_,_).

  longmenu1(cont,ROW,COL,H,W,ATTR,STRINGLIST,NOOFROW,OLDBASE,BASE,OFFS,BASE2,OFFS2) :-
        scr_box(add),
        wr_part_if_changed(OLDBASE,BASE,H,W,STRINGLIST),
        crt(Crt),
        set_attr(Crt,reverse,ATTR,REV),
        field_attr(OFFS,0,W,REV),
        cursor(OFFS,0),shiftwindow(command_window),
        readkey(KEY),
        longmenu2(STOP,H,NOOFROW,BASE,OFFS,BASE1,OFFS1,KEY),
        shiftwindow(menu_window),
        field_attr(OFFS,0,W,ATTR),
        scr_box(remove),
        No=BASE1+OFFS1+1,write command_hint(No,STRINGLIST).!.
        longmenu1(STOP,ROW,COL,H,W,ATTR,STRINGLIST,NOOFROW,BASE,BASE1,OFFS1,BASE2,OFFS2).
  longmenu1(esc,_,_,_,_,_,_,_,_,OB,OF,OB,OF):-!.
  longmenu1(_,_,_,_,_,W,ATTR,_,_,_,OB,OF,OB,OF):-
        shiftwindow(menu_window),crt(Crt),!,
        set_attr(Crt,reverse,ATTR,REV),
        field_attr(OF,0,W,REV).
```

```
longmenu2(esc,_,_,_,_,0,-1,esc)          :-!.
longmenu2(esc,_,_,_,_,0,-1,fkey(10))     :-!.
longmenu2(cont,_,_,B,O,B,O,fkey(1))      :-!,help.
longmenu2(selection,_,_,B,O,B,O,cr)      :-!.
longmenu2(cont,_,_,_,_,0,0,home)         :-!.
longmenu2(cont,H,LEN,_,_,B1,O1,end)      :-!, O1=H-1, B1 = LEN-H.
longmenu2(cont,H,LEN,B,O,B1,O1,up)       :-!, CO=O-1,longmenu3(LEN,H,B,CO,B1,O1).
longmenu2(cont,H,LEN,B,O,B1,O1,down)     :-!, CO=O+1,longmenu3(LEN,H,B,CO,B1,O1).
longmenu2(cont,H,LEN,B,O,B1,O1,pgup)     :-!, CO=O-H+1,longmenu3(LEN,H,B,CO,B1,O1).
longmenu2(cont,H,LEN,B,O,B1,O1,pgdn)     :-!, CO=O+H-1,longmenu3(LEN,H,B,CO,B1,O1).
longmenu2(cont,_,_,B,O,B,O,ins)          :-!, changemode.
longmenu2(cont,_,_,B,O,B,O,char(K)):-
      insmode,shiftwindow(command_window),
      cursor(Row,Col),d_col(_,Max),
      Len=Max-Col-2,Len>0,!,field_str(Row,Col,Len,String),C1=Col+1,
      write(K),write(String),cursor(Row,C1).

longmenu2(cont,_,_,B,O,B,O,char(K)):-
      shiftwindow(command_window),cursor(_,Col),
      d_col(_,Max),Col<Max-1,!,write(K).
longmenu2(cont,_,_,B,O,B,O,right):-
      shiftwindow(command_window),cursor(Row,Col),Col1=Col+1,
      d_col(_,Max),Col1<Max,!,cursor(Row,Col1).
longmenu2(cont,_,_,B,O,B,O,left):-
      shiftwindow(command_window),cursor(Row,Col),Col1=Col-1,
      d_col(Min,_),Col1>Min,!,cursor(Row,Col1).
longmenu2(cont,_,_,B,O,B,O,del):-
      shiftwindow(command_window),cursor(Row,Col),d_col(_,Max),!,
      C1=Col+1,Len=Max-C1,field_str(Row,C1,Len,String),
      write(String),cursor(Row,Col).
longmenu2(cont,_,_,B,O,B,O,bdel):-
      shiftwindow(command_window),cursor(Row,Col),d_col(Min,Max),
      C1=Col-1,C1>Min,!,Len=Max-Col,field_str(Row,Col,Len,String),
      cursor(Row,C1),write(String),cursor(Row,C1).
longmenu2(cont,_,_,B,O,B,O,_).

longmenu3(_,H,B,O,B,O)                   :-O>=0, O<H, !.
longmenu3(LEN,H,B1,O1,B2,O2)  :-O1+B1>=LEN,!, O2=H-1, B2=LEN-H.
longmenu3(_,_,B1,O1,0,0)                 :-O1+B1<0,!.
longmenu3(_,_,B1,O1,B2,0)     :-O1<0, !, B2=B1+O1.
longmenu3(_,H,B1,O1,B2,O2)    :-O1>=H, O2=H-1, B2=B1+O1-O2.

wr_part_if_changed(B,B,_,_,_)  :- !.
wr_part_if_changed(OLDB,B,H,W,CL) :-
     SCROLL=B-OLDB, scroll(SCROLL,0),
     write_part_list(0,B,0,H,W,CL).
write_part_list(_,_,H,H,_,_) :- !.
write_part_list(I,B,R,H,W,[_|T]) :- I<B,!, I1=I+1,
            write_part_list(I1,B,R,H,W,T).
write_part_list(I,B,R,H,W,[STR|T]) :-
            shiftwindow(menu_window),
            concat(" ",STR,STR1),
            field_str(R,0,W,STR1),
            R1=R+1,
            write_part_list(I,B,R1,H,W,T).

changemode:-retract(insmode),!.
changemode:-assert(insmode).
```